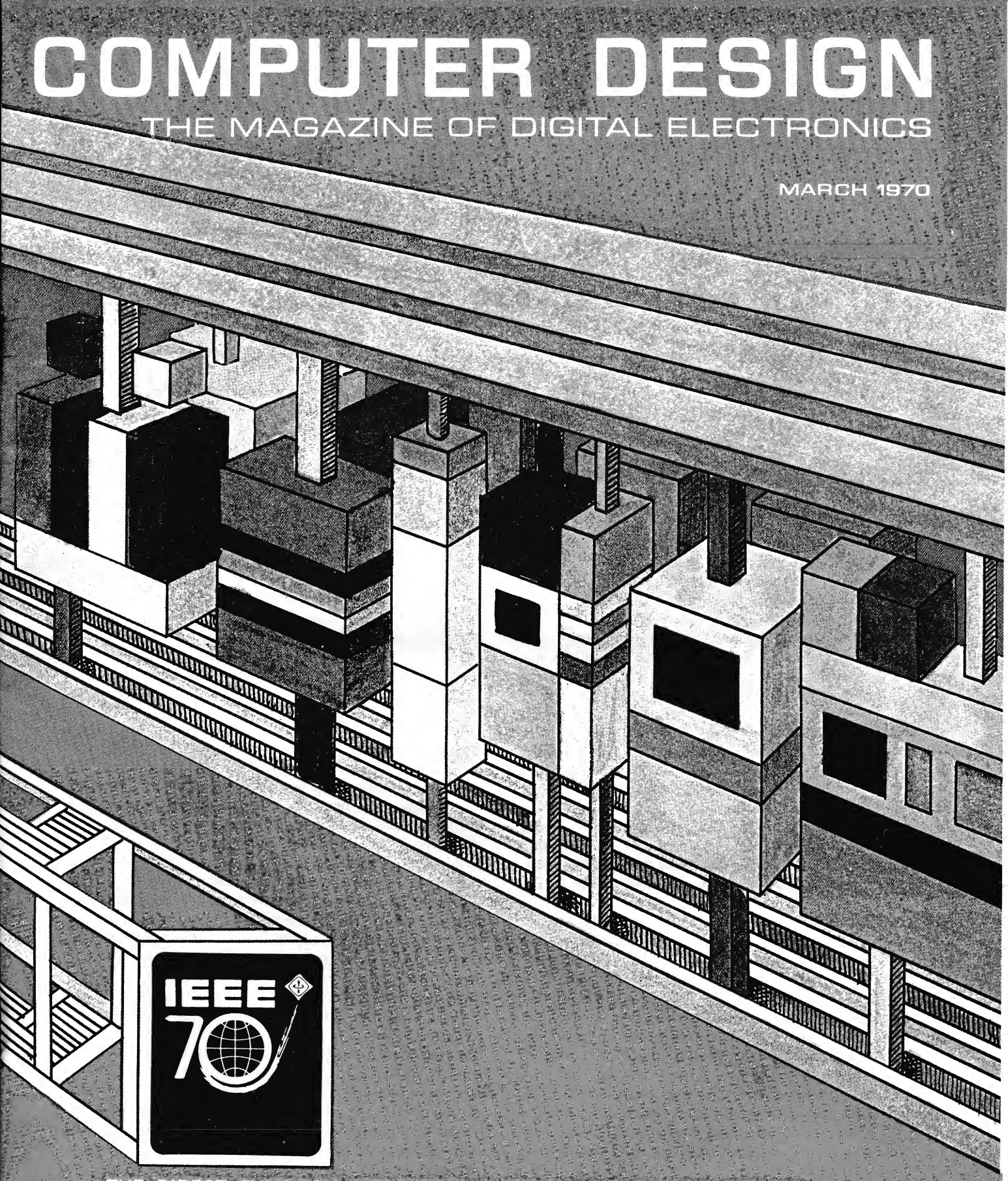
COMPUTER DESIGN
THE MAGAZINE OF DIGITAL ELECTRONICS
MARCH 1970




# THE DIRECT FUNCTION PROCESSOR CONCEPT FOR SYSTEM CONTROL

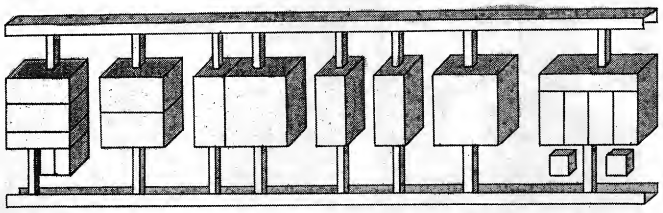
**Saul B. Dinman**

GRI Computer Corporation
Newton, Massachusetts

REPRINTED FROM COMPUTER DESIGN, MARCH 1970

*Here is an inexpensive approach to system control when arithmetic manipulation is not a prime factor. The direct function processor is a functionally oriented machine having true modularity and expandability*

# The Direct Function Processor Concept for System Control



**Saul B. Dinman**

GRI Computer Corporation
Newton, Massachusetts

At the present time, many small general purpose computers are being used as systems controllers. However, the fundamental concept of computers is that the computer is primarily a calculating device. It was never conceived in terms of controlling and sequencing activities that might not involve mathematics. The systems designer is forced to program his system functions in arithmetic terms understood by the computer but which might bear little direct relationship to its functional use.

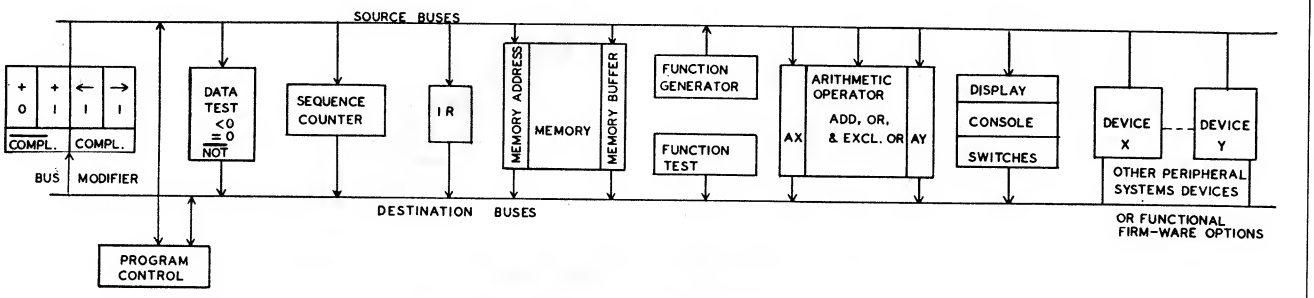
A fact generally overlooked about general purpose computer-controllers is that they are arithmetically, not functionally, oriented. All have a central processing unit (CPU) that is designed to accumulate information from a system or subsystem, perform arithmetic calculations on it, and return the computed data to the outside world, where it can be used to instruct, adjust, correct, regulate, or protect the system that is being controlled.

The cost and capability of these machines depend on how clever the computer manufacturer has been in designing and interconnecting his accumulators, core memories, arithmetic units, instruction and index registers, and other key elements. In general, all the computers that have been developed to date differ only in the design and organization of these elements. The concept has been that of calculating devices, not of control and sequencing devices, and, obviously, the objective is to obtain the most computing power per dollar that is possible.

The arithmetic concept requires either that the computer be required to perform a very specific function or that it be a general purpose machine with some predetermined capability-to-cost ratio based on what the computer manufacturer believes a group of users will require in terms of power and on what peripheral equipment they will connect to the input/output lines. Since the obvious engineering/sales decision is to aim for the broadest market, the systems designer usually ends up paying more for his machine than is warranted. In short, the progression has been from a single purpose computer-controller to a general general computer-controller—indeed, one that is often so general that it can be justified only if it is applied for multiple purposes, such a production control and accounting, as well as systems control.

Furthermore, every time there is a breakthrough that provides more power per dollar (which of course is associated not only with improved arrangements of the components inside the computer, but with advances in solid state devices, integrated circuits, packaging, component design, and the like) the industry moves to a different magnitude of capability and to

***Saul B. Dinman is company founder and vice-president of engineering. Concepts developed by him and implemented in the direct function processor are the result of many years of experience in logic design, programming, and computer controlled systems. Mr. Dinman received a BSEE degree from Pennsylvania State University and is a professional engineer.***




Typical configuration of direct function processor

the so-called next generation machine that makes obsolete the user's current machine. To put it mildly, the systems designer is not an enthusiastic supporter of this cycle. It forces him to redesign his control system in order to remain competitive and puts him on an economic treadmill that can jeopardize his very existence. In existing general purpose computers there is neither true functional modularity, expandability, nor upward or downward software compatability between family members—despite what the industry claims.

Another disadvantage of this arithmetic concept is that the input data must be programmed in terms that the computer can understand, and the output data, in terms that the system can understand. Thus the systems designer is forced to program his system functions in arithmetic terms understood by the computer, and these unfortunately bear little direct relationship to its functional use. In fact, in most cases the actual systems programming is turned over to a programmer who must try to relate the systems requirements to equivalent terms that can be translated as instructions to the computer. Frequently, the programmer does not fully comprehend the total aspects of the system he is programming. In a sense the systems designer in turning the programming responsibility over to the programmer loses control of the design of his own system.

As a result, programming services have risen sharply in cost as the demand has increased. A new industry has grown up providing programming services to systems people because the demand far exceeds the supply—and the situation is worsening rapidly.

Another serious drawback of most of the current computer-controller systems is that in many cases—perhaps the majority—neither the computer manufacturer nor the system's producer is capable of maintaining the other's equipment. And while the user understands and can maintain the functional system, he does not have the same competence when it comes to the computer. So the computer manufacturer must support an expensive group of trouble-shooting and maintenance specialists that duplicates the field staff of the OEM supplier. In the final analysis, it's the customer who pays.

## THE DIRECT FUNCTION PROCESSOR CONCEPT

The GRI-909 computer, or direct function processor, differs from the conventional computer-controller in that it extends into the computer the bus system across which the control devices are normally connected. To put it more accurately, the elements of the computer are brought outside the central processor, and all computer devices such as the instruction register, sequence counter, arithmetic unit, and memory, are connected across the same bus structure as the devices in the equipment to be controlled.

Thus, all internal and external elements are directly addressable by use of a compiler-like functional language rather than a mathematically oriented language. The programmer does not have to develop the involved command sequences that will translate process data into a language that the computer can understand and then back again into instructions that the equipment or process can react to. The unimpeded flow of data from device to device saves temporary storage locations for "bookeeping" purposes and provides the designer with modularity and flexibility to adapt the computer to specific system requirements.

The key to the concept of the direct function processor is a block of logic that provides a programmable path between the source and destination buses in the computer. It is termed a bus modifier and is designed to take information from any input device and move it to any output device, performing operations on the data as it passes from one device to the other.

Since the path connection is programmable, data can be transmitted from one device to another in one of the following conditions: unchanged, incremented, shifted left one bit, shifted right one bit, "ones" complemented, or "ones" not complemented. A "twos" complement, or negative number, can be obtained on the fly by combining the "ones" complement capability with the increment capability.

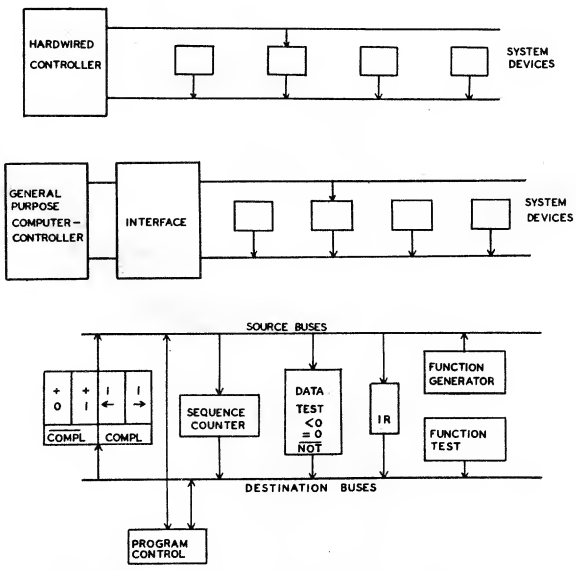
A link bit, through which data can be shifted for testing one bit at a time, and an overflow bit for tests involving incrementing data are provided with the bus modifier. Thus the systems designer has the ability to count and control data, although restricted to simple, repetitive operations, on a bit-by-bit basis.

## INSTRUCTION FORMAT

The GRI-909 is a 16-bit, parallel machine. Six of the bits are used for source addresses, six for destination addresses, and four to control the transfer of data throughout the system. The latter can be microprogrammed to modify data or control their transfer.

Each device in the system is equipped with a decoder that enables it to recognize an instruction to supply, receive, modify, or otherwise manipulate data. Thus, in effect, the control logic is spread throughout the machine and is tailored to the particular requirements of the equipment to be controlled, since it is only added to the system when another device is added.

Program control provides the signals that indicate when a source device is to give up data, sets up the route the data are to follow, and tells the source device that is to receive them. It also must be capable of subsystem manipulation: incrementing the sequence counter or changing its contents when a jump occurs. A read-only memory, whose instructions are in the same format and therefore indistinguishable from those coming from the instruction register, is used to enable program control to perform such manipulations over the same data and control structure as used for the movement of system data.




Circuit arrangements for typical controller, general purpose computer, and tie-in system

## BASIC COMPUTER DEVICES

The devices connected between source and destination buses in the basic computer configuration are, in addition to the bus modifier, the following.

Sequence Counter: This device is provided to keep track of the program information. It is common to all computers and indicates the address of the next instruction. In this application, the sequence counter is connected across the buses, as are all other elements in the system, providing direct access from device to device. An external device can cause the program to go to some special subroutine in memory by transmitting an address word directly to the sequence counter.

Instruction Register: The instruction register contains the instruction in the computer to be executed at a given moment. Like other elements in this system organization, it is connected across the source and destination buses.

Data Test: A computer decides on the program paths it will follow on the basis of the value of the data that resides in the arithmetic structure. In this machine, data test determines whether the value of the information it receives from any source is less than zero, equal to zero, or any combination thereof (including the negation). This tester is connected between the source and destination buses and is programmed to accept data directly from any source. A positive response to a data test results in a jump instruction. The contents of the sequence counter are automatically stored in a trap register associated with data test when a jump is executed.

Function Generator: Most peripheral devices require control pulses to perform such functions as start, stop, clear, and the like. Up to 16 different control commands can be issued to each system device by generating the address of the device and the control pulses on four control lines provided for this purpose. A typical instruction to be issued by the function generator might be START READER.

Function Test: Some devices produce status signals which indicate certain conditions to the computer. The function test operator looks at these status signals and acts upon them. Three control lines are provided for this purpose, plus a fourth which provides logical negation of the other three. A positive response by the function test operator to the sense lines results in a skip instruction.

Console: If a peripheral device is added to a system, a set of lights may be required to indicate what is going on within the device. When the switches on the console are set to a device address, any data delivered to that device will be displayed. This is useful for maintenance and debugging purposes. In some systems this can eliminate the development of large display panels and their attendant cost. The contents of any source register in the system can also be displayed on the computer console.

A programmer's console is optionally available which simultaneously displays major internal registers in the computer in addition to the selectable data display. Both consoles are equipped with plugs and are interchangeable.

Data may also be transmitted to any device in the system from the console switches by selecting its address and activating the transmit key. Control switches on the console are: start, continue, read, write, display, transmit, single step, and stop.

Minimum Configuration: The system described so far is the minimum configuration for the computer. With the addition of some kind of stored program, it can provide all the control capabilities of a general purpose computer. It can test data, transmit and receive control signals, and perform arithmetic and logic op-

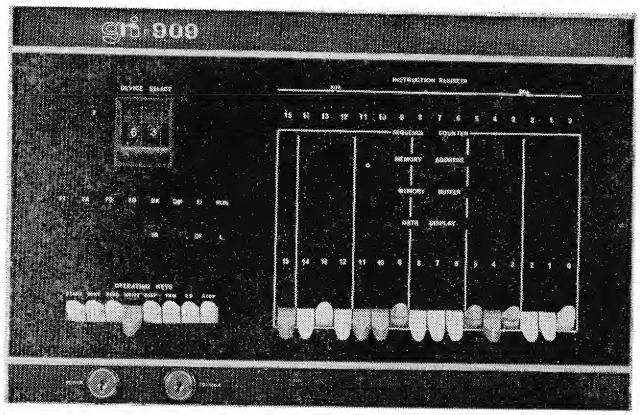
erations one bit at a time as data pass through the bus modifier. Although limited in the execution times of its arithmetic operations, it can be used as a special purpose controller with read-only memory or a general purpose controller with random access memory for those applications requiring little or no arithmetic or when the execution times of arithmetic are not a critical factor. This provides a very inexpensive approach to system control, with the capacity for expansion to full computer capability if required.

Core Memory: The memory is a 16-bit random access, ferrite core memory in 1024- and 4096-word plug-in modules. It can be expanded to 8192 in the basic processor frame without additional wiring. The total expansion capability is 32,768 directly addressed words.

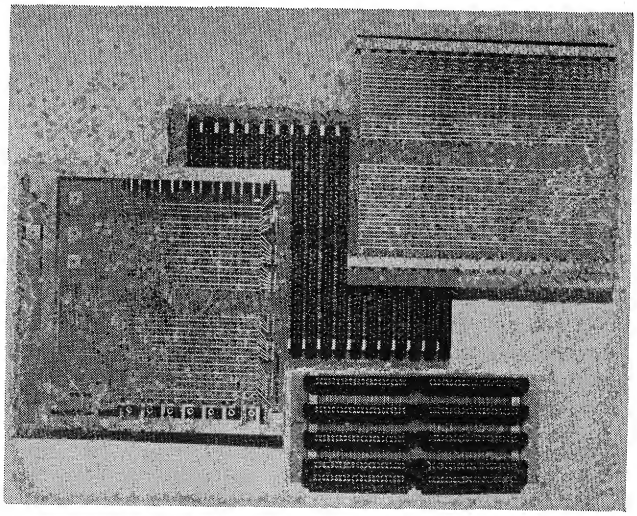
There is a multiple-channel, single-cycle, direct memory access system in the basic processor that permits direct, single-word access to the memory by a system function. The difficulty of multiplexing several devices on many different data channels is eased by a simple priority allocation system. Similarly, a range of complexity in implementation is available to the system designer. The data channels may be used to transfer data in or out of memory or simply to increment a specified memory location.

Arithmetic Operator: The arithmetic and logic manipulations that can be performed in the functional arithmetic operator are "add," "and," "or," "and/or," and "exclusive or." This arithmetic operator works somewhat differently from that of a typical computer. Instructions are not issued that say "add," which in a conventional computer says, "one number is in the accumulator and the other number is in memory. Pull the number out of memory, add the two together, and put the sum back into the accumulator."

In the direct function processor, the function generator is used to generate the "add" function. The instruction looks like "Function output 'add' to the arithmetic operator." This element will always perform the "add" function between the current value of X and Y accumulators until the user issues another command changing the state. When either one of those registers is changed, a new sum appears, immediately available for transfer to any point in the system. New values can be presented from a system register with a new result obtained in a single cycle time of 1.76 microseconds. The result, available as a separate source, always reflects the instantaneous output generated by the contents of the X and Y accumulators as controlled by the function selected. It can be stored in memory by a single instruction. The introduction of new values to one accumulator does not alter the contents of either accumulator, unless the instruction "arithmetic operator to X (or Y) accumulator" is issued.




The GRI-909 computer



Computer internal wiring

## FIRMWARE OPERATORS

Computer capabilities can be expanded by adding firmware operators to the system. A firmware operator generally presents at least one register to the bus system. Each functional operator added to the computer systems adds one or more hardware instructions to the computer plus the variations provided by the bus modifier. Examples of some of the firmware options are the following:

Multipler: A "multiply" can be executed either through subroutine instructions or by the multiply operator, a firmware option. The subroutine for "multiply" has a maximum execution time of 360 microseconds for a full 31-bit signed product and occupies 42 memory locations.

The multiply operator performs a 16-bit unsigned multiplication. It uses the arithmetic operator and issues external instruction requests to the processor. It uses a single address for the 16-bit register (MPR) housed in the operator, which may also be used as a temporary storage register.

The 31-bit product is available in the X-accumulator of the arithmetic operator (most significant) and MPR (least significant) 56.32 microseconds after starting the operator.

The execution time might be extended if direct memory access requests arrive during the processing of this instruction. A higher-speed multiply operator is also available with an execution time of under 10 microseconds.

Byte Swap: The byte swap subroutine requires 20 memory locations and executes in 124 microseconds.

The byte swap operator is a 16-bit register (SWAP) used to interchange the halves of a computer word. It executes the interchange in a single cycle time of 1.76 microseconds, exchanging bits 15 to 8 with bits 7 to 0. A byte pack operator is also available to form a 16-bit word from two 8-bit characters that are loaded sequentially.

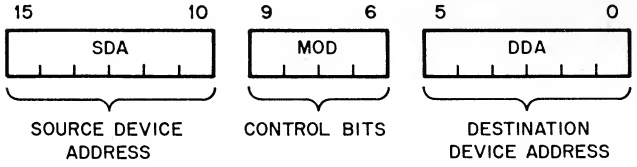
These two examples are indicative of the flexibility built into the computer. The ability is intrinsic to the direct function processing technique, permitting virtually any operation to be incorporated in the instruction repertoire of the machine. Other standard options include: square root, BCD/binary conversion, and general purpose registers.

## DEVICE OPERATORS

Device operators provide the interface circuitry between system devices and the computer. Direct memory access and priority interrupt circuitry are provided for each device, as required, in the device operator. The high-speed signals related to the internal operation of the computer are terminated at the device operators so that external devices see only relatively slow signals that can be cabled to the device. In some cases, if the device is simple, the entire device can be provided on the device operator plug-in module. This module is approximately 4 inches by 9 inches and provides an output cable connection.

## PROGRAMMING

Instructions are of the general form DEVICE X TO DEVICE Y, and are described by a single machine format:

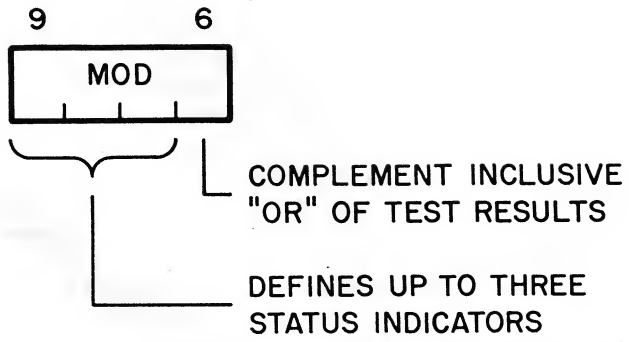



The actual operation performed by an instruction is dependent upon the unique combination of SDA, MOD, and DDA.

The four functional instruction classes are outlined below. Each class is described with examples of machine and corresponding functional language instructions in the following paragraphs. In machine language SDA and DDA are represented as two octal digits each, and MOD is represented as four binary digits.

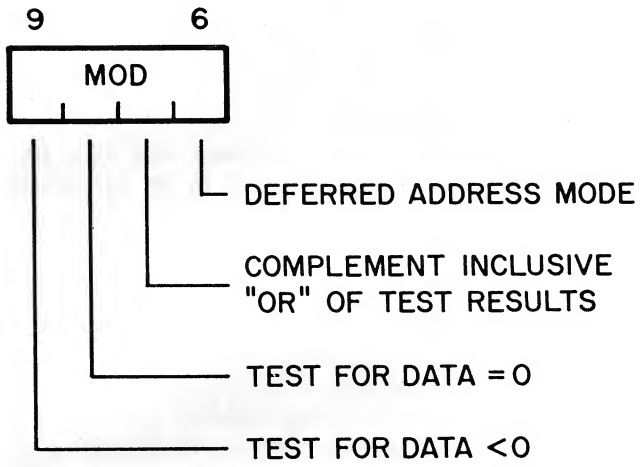
Function Generator: The source address of the function generator is 02. It causes control signals (rather than data) to be transmitted to any system device specified by the DDA. The modifier defines up to four pulses in combination to be transmitted in parallel. Examples of FAST instructions are: START HSR START TTI.

Function Test: The destination address of function test is 02. It senses up to three status indicators associated with any system device as specified by the source address. If the status test defined by the modifier is true, a SKIP instruction is performed. The format for the modifier is




Examples of FAST instruction: SKIP IF ADC RDY SKIP IF TTI NOT RDY

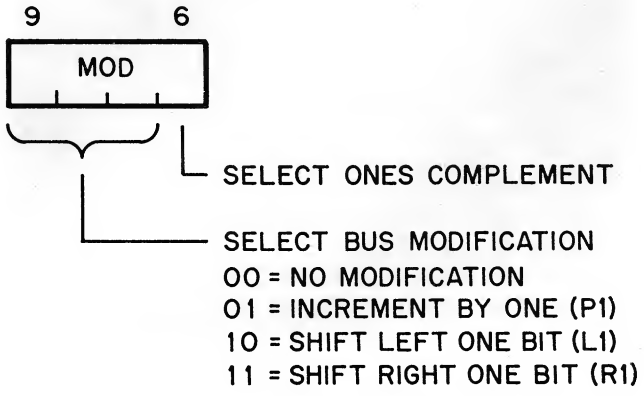
Data Test: The destination address of data test is 03. It can test data from any source address for less than zero, equal to zero or any combination thereof. If the test defined by the modifier is true, the next word is taken as a jump address; otherwise, the next word is skipped. The format for the modifier is



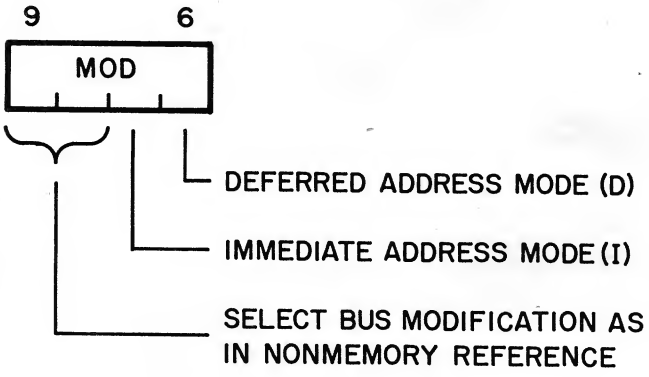

If a jump is performed, the current value of the sequence counter is automatically stored in a trap register associated with the data test. If the jump is to a subroutine, the trap register provides the link back to the calling program.

Examples of FAST instructions: IF ADC ETZ GO TO ALARM. IF TTI NETZ GO TO PROCESS

Data Transmission: Any instruction not in one of the classes defined above implies the transmission of data from a source device to a destination device as specified by their addresses. The source and destination may be the same device. Data transmission instructions are of two forms. For nonmemory reference the modifier format is



Examples: AO P1 TO DAC. C ADC P1 TO AY
For memory reference a memory reference instruction is designated if the source or destination address is the memory buffer register (06). The next word is taken as a memory address or an operand. The format for the modifier is




Examples: I-27 TO AX. LIMIT TO AY. D LIST-1 TO DAC

The contents of any register connected to the computer bus structure can be transferred to any other register in the system by a single instruction. The registers can be any devices in the computer or in the system, including memory. The computer recognizes and services interrupt requests and direct memory requests generated by system devices. Devices may, however, cause traps to unique memory locations. The cost of generating a unique address is borne by the device interface itself. In a multilevel interrupt environment, priority is enforced by manipulations of an interrupt status register by the respective interrupt routines. Each device that is interfaced to operate in an interrupt mode contributes a bit to the interrupt status register so that it may be selectively enabled and disabled. Interrupts may cause a trap to memory location zero. If two or more devices trap to the same memory location, a standard SKIP sequence is used by the interrupt executive routine to enforce priorities and isolate the device, or the devices may generate two unique addresses.

Input/output devices may be driven in a programmed synchronized mode, normal interrupt mode, or direct memory access mode. The permissible mode(s) for a device is determined by its interface with the system, but all data connections are made via the same bus structure used for programmed transfer.

A device to be operated in the direct memory mode presents at least two registers to the bus system (memory starting address and data buffer). A typical direct access device interface will appear to the user as follows:
Three addresses—DSKA: starting address on disc (prime register); DSKM: starting address in memory; DSKL: block length to be transferred.
Example: read from disc
TRACK TO DSKA : SET DISK ADDRESS
BUFFA TO DSKM : SET CORE ADDRESS
LENG TO DSKL : SET LENGTH
READ TO DSKA : INITIATE READ

The device will steal one machine cycle (1.76 microseconds) for transfer of each data word. When the block transfer is completed, the device generates an end-of-range interrupt on the priority interrupt channel. In addition, the disc contributes one bit to the machine status word to indicate its active state and sense flags to indicate parity error and up-to-speed.

Another interrupt mode, external instruction request, is provided. This interrupt mode is used to exercise the computer with hardware diagnostic modules. It can also be used in conjunction with read-only memory to extend the instruction repertoire to include complex instructions. When read-only memory is used in this interrupt mode, an instruction is executed in 880 nanoseconds.

## SUMMARY

The computer is both modular and expandable. It need not have a main memory since other functions can be added later. Or it may have a minimal amount of memory and arithmetic capability, these functions being substantially expanded for different control system models. Thus the system designer can evaluate his system requirements with regard to the trade-offs between speed and economy. If he is looking for economy, he can strip down his hardware and perform his system functions through program subroutines. If it is speed he is looking for, he can substitute hardware with an increased cost.

The firmware capability is intrinsic to the computer. It therefore provides the flexibility of adding the firmware option at any time. If requirements change after a system has gone into operation, it is possible to expand the computer capability. So a decision made at one time does not necessarily commit the system designer for all time or require substitution of a different, more powerful processor.

Certain special hard-wired instructions that cannot be found in any other small computer or, indeed, in any other computer, small or large, can be provided or added by the customer as his own proprietary functions. For example, in a system monitoring fluid flow, the square root of the output of the flow-meters might be necessary for normalizing the results. A hard-wired square root instruction is possible. This is not an instruction capability normally found in small computers.

Another example might be a machine tool numerical control system. The system manufacturer might develop some special interpolation technique. By incorporating this within the computer he has a proprietary and unique computing capability. No other computer will be able to exactly duplicate that capability. In fact, this is an excellent way for the system manufacturer to maintain his identity and image where there is a growing tendency for one computer system to resemble another. In short, virtually any type of system dependent upon automated control can be enhanced by direct function processing.

GRI Computer Corporation
320 Needham Street, Newton, Massachusetts 02164 Tel. (617) 969-7346

# NORTH AMERICAN SALES OFFICES

## U. S. NORTHEAST REGION

New England States

Regional Sales Office:

Les E. Silvern
320 Needham Street
Newton, Mass. 02164
Call: (617) 969-7346

NACO ELECTRONICS CORP.

P. O. Box 248
South Bay Road
North Syracuse, New York 13212
Call: (315) 699-2651

74 Park Avenue
Rochester, New York 14607
Call: (716) 244-4720

2631 Edgewood Road
Utica, New York 13501
Call: (315) 732-1801

1032 Merlin Drive
Schenectady, New York 12309
Call: (518) 785-3750

Buffalo, New York
Call: (716) 836-5108

Binghamton, New York
Call: (607) 724-8288

Poughkeepsie, New York
Call: (914) 462-2730

## U. S. EASTERN REGION

Regional Sales Office

Jim Greene
1080 Route 46
Clifton, New Jersey 07013
Call: (201) 773-4646

RON DAVIES ASSOCIATES, INC.

10 Old Post Office Road
Silver Springs, Maryland 20910
Call: (301) 587-3910

P. O. Box 5146
Charlottesville, Virginia 22903
Call: (703) 293-6701

REYNOLDS AND ASSOCIATES, INC.

822 East Strawbridge Avenue
Melbourne, Florida 32901
Call: (305) 727-3205

412 West Market Street
Greensboro, North Carolina 27401
Call: (919) 275-3725

175 W. Wieuca Road, N. E.
Suite 139
Atlanta, Georgia 30305
Call: (404) 252-5360

904 Bob Wallace Avenue, S. W.
Suite 103
Huntsville, Alabama 35801
Call: (205) 536-1941

161 S. W. 75th Avenue
Ft. Lauderdale, Florida 33314
Call: (305) 581-6611

P. O. Box 8877
Orlando, Florida 32806
Call: (305) 843-6440

## U. S. CENTRAL REGION

Regional Sales Office

Raymond L. Baker
3774 Oak Forest Drive
Toledo, Ohio 43614
Call: (419) 385-8723

SEA, INC.

4210 W. Irving Park Road
Chicago, Illinois 60641
Call: (312) 282-6694

100 W. 96th Street
Suite 3A
Minneapolis, Minnesota 55420
Call: (612) 888-1969

420 W. Ravenswood Hills
Waukesha, Wisconsin 53186
Call: (414) 782-3388

1010 East 86th Street
Indianapolis, Indiana 46240
Call: (317) 846-2593

THE THORSON COMPANY

7700 Carpenter Freeway
Dallas, Texas 75247
Call: (214) 631-5440

6655 Hillcroft - Suite 224
Houston, Texas 77036
Call: (713) 771-3504

## U. S. WESTERN REGION

INSTRUMENT SPECIALISTS, INC.

GRI Computer Specialist
Jerry Satuloff
1109 South Central Avenue
Glendale, California 91204
Call: (213) 245-9404

2359 De La Cruz Boulevard
Santa Clara, California 95050
Call: (408) 244-1505

10459 "A" Roselle Street
San Diego, California 92121
Call: (714) 453-1833

INTER-LINK SYSTEMS

8345 West 16th Avenue
Lakewood, Colorado 80215
Call: (303) 237-7055

Rte. 4 - Box 4533
Bainbridge Island, Washington 98110
Call: (206) 634-1717

## CANADA

INSTRONICS TECHNO-PRODUCTS LTD.
P. O. Box 100
Stittsville (Ottawa) Ontario - CANADA

Call:
Ottawa (613) 836-4411
Montreal (514) 861-1375
Toronto (416) 536-7000
Vancouver (604) 688-2619